IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: M. Ueda

Date: September 6, 2001

Serial No.: 09/682,188

Docket No.: JP920000046US1

Filed: August 2, 2001

Group Art Unit: 2185

FOR: Multiprocessor System, Processor Module For Use Therein, And Task Allocation Method In Multiprocessing

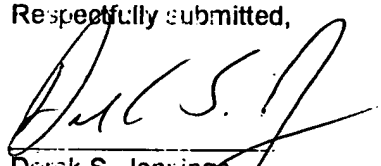
Assistant Commissioner for Patents
Washington, D.C. 20231

SUBMISSION OF PRIORITY DOCUMENT

Sir:

Enclosed herewith is a certified copy of Japanese Application No. 2000-236740 filed August 4, 2000 , in support of applicant's claim to priority under 35 U.S.C. 119.

Respectfully submitted,



Derek S. Jennings
Reg. Patent Agent/Engineer
Reg. No.: 41,473
Tel. No.: (914) 945-2144

IBM CORPORATION
Intellectual Property Law Dept.
P. O. Box 218
Yorktown Heights, N. Y. 10598

00 046
YOR
#5

日　本　国　特　許　庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２０００年　８月　４日

出願番号
Application Number: 特願２０００－２３６７４０

出願人
Applicant(s): インターナショナル・ビジネス・マシーンズ・コーポレーション

２００１年　５月３１日

特許庁長官
Commissioner,
Japan Patent Office 及川耕造

出証番号　出証特２００１－３０５００７１

【書類名】　特許願
【整理番号】　JP9000046
【あて先】　特許庁長官殿
【国際特許分類】　G06F 12/08
【発明者】
【住所又は居所】　滋賀県野洲郡野洲町大字市三宅８００番地　日本アイ・ビー・エム株式会社　野洲事業所内
【氏名】　上田　真
【特許出願人】
【識別番号】　390009531
【氏名又は名称】　インターナショナル・ビジネス・マシーンズ・コーポレーション
【代理人】
【識別番号】　100086243
【弁理士】
【氏名又は名称】　坂口　博
【電話番号】　0462-15-3318
【復代理人】
【識別番号】　100094248
【弁理士】
【氏名又は名称】　楠本　高義
【選任した代理人】
【識別番号】　100091568
【弁理士】
【氏名又は名称】　市位　嘉宏
【手数料の表示】
【予納台帳番号】　012922
【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　9706050

【包括委任状番号】　9704733

【プルーフの要否】　要

【書類名】　　明細書

【発明の名称】　マルチプロセッサ・システム、マルチプロセッサ・システムに用いるプロセッサ・モジュール及びマルチプロセッシングでのタスクの割り当て方法

【特許請求の範囲】

【請求項１】　１つのキャッシュ・メモリとこのキャッシュ・メモリを共有する複数のプロセッサとを含むプロセッサ・モジュールを複数用いたマルチプロセッシングでの各プロセッサへのタスクの割り当て方法であって、

複数のプロセッサ・モジュール内のキャッシュ・メモリに記憶されている共通データへの各タスクのアクセス状況を監視するステップと、

前記アクセス状況に基づいて、同一の共通データへのアクセス回数が多いタスクを、同一プロセッサ・モジュール内のプロセッサに割り当てるステップと

を含むタスクの割り当て方法。

【請求項２】　前記アクセス状況を監視するステップが、

他のプロセッサ・モジュールのキャッシュ・メモリにも記憶されている共通データを書き換えたことによる、前記他のプロセッサ・モジュールのキャッシュ・メモリ内の共通データの無効化の発生を検出するステップと、

前記無効化を発生させたタスクの識別情報と、書き換えられたデータのアドレスと、同一タスクによる同一データでの無効化発生回数とを記憶するステップとを含み、

前記プロセッサに割り当てるステップが、

同一データでの無効化発生回数の多さに基づいて、タスクをグループ分けするステップと、

同一グループに分けられた各タスクを同一プロセッサ・モジュール内の各プロセッサに割り当てるステップと

を含む請求項１のタスクの割り当て方法。

【請求項３】　前記アクセス状況を監視するステップが、

複数のキャッシュ・メモリに記憶されている共通データのいずれかが書き換えられたことによる、前記共通データのうちの前記書き換えられたデータ以外のデ

ータの無効化を検出及び記憶するステップと、

前記無効化されたデータへの各タスクのアクセス状況を監視するステップと

を含み、

前記プロセッサに割り当てるステップが、

同一の無効化されたデータへのアクセス回数の多さに基づいて、タスクをグループ分けするステップと、

同一グループに分けられた各タスクを同一プロセッサ・モジュール内の各プロセッサに割り当てるステップと

を含む請求項１のタスクの割り当て方法。

【請求項４】　前記無効化されたデータへの各タスクのアクセス状況を監視するステップが、

無効化されたデータへのアクセスを検出するステップと、

アクセスが検出された無効化されたデータのアドレスと、この無効化されたデータへアクセスしたタスクの識別情報と、同一タスクによる同一アドレスへのアクセス回数とを記憶するステップと

を含む請求項３のタスクの割り当て方法。

【請求項５】　前記無効化されたデータへのアクセスを検出するステップが、

データの無効化を検出するステップと、

無効化されたデータのアドレスを記憶するステップと、

キャッシュ・ミスを検出するステップと、

キャッシュ・ミスが発生したデータ・アドレスと前記記憶された無効化されたデータのアドレスとを比較するステップと

を含む請求項４のタスクの割り当て方法。

【請求項６】　前記記憶された各無効化発生回数又は各アクセス回数の合計値が所定値を超えると、前記各プロセッサへのタスクの割り当てを、マルチプロセッシングを行っているオペレーティング・システムに要求するステップを含む請求項２、請求項４又は請求項５のタスクの割り当て方法。

【請求項７】　１つのキャッシュ・メモリとこのキャッシュ・メモリを共有

する複数のプロセッサとを含むプロセッサ・モジュールを複数用いたマルチプロセッサ・システムであって、

（ａ）複数のプロセッサ・モジュール内のキャッシュ・メモリに記憶されている共通データへの各タスクのアクセスを検出する手段と、

アクセスが検出された共通データのアドレスと、この共通データへアクセスしたタスクの識別情報と、同一タスクによる同一データへのアクセス回数とを記憶する記憶手段と

を含むプロセッサ・モジュールと、

（ｂ）前記アクセス回数に基づいて、同一の共通データへのアクセス回数が多いタスクを、同一プロセッサ・モジュール内のプロセッサに割り当てる手段と

を含むマルチプロセッサ・システム。

【請求項８】　前記プロセッサ・モジュールのアクセスを検出する手段が、

他のプロセッサ・モジュールのキャッシュ・メモリにも記憶されている共通データを書き換えたことによる、前記他のプロセッサ・モジュールのキャッシュ・メモリ内の共通データの無効化の発生を検出する手段と、

前記無効化を発生させたタスクの識別情報と、書き換えられたデータのアドレスと、同一タスクによる同一データでの無効化発生回数とを記憶する手段と

を含む請求項７のマルチプロセッサ・システム。

【請求項９】　前記プロセッサ・モジュールのアクセスを検出する手段が、

複数のキャッシュ・メモリに記憶されている共通データのいずれかが書き換えられたことによる、前記共通データのうちの前記書き換えられたデータ以外のデータの無効化を検出及び記憶する手段と、

前記無効化されたデータへの各タスクのアクセスを検出する手段と

を含む請求項７のマルチプロセッサ・システム。

【請求項１０】　前記データの無効化を検出及び記憶する手段が、

データの無効化を検出する手段と、無効化されたデータのアドレスを記憶する手段とを含み、

前記無効化されたデータへの各タスクのアクセスを検出する手段が、

キャッシュ・ミスを検出する手段と、キャッシュ・ミスが発生したデータ・ア

ドレスと前記記憶された無効化されたデータのアドレスとを比較する手段とを含む請求項９のマルチプロセッサ・システム。

【請求項１１】　前記記憶された各アクセス回数又は各無効化発生回数の合計値が所定値を超えると、前記各プロセッサへのタスクの割り当てを、前記割り当て手段に要求する手段をさらに含む請求項７又は請求項８のマルチプロセッサ・システム。

【請求項１２】　マルチプロセッサ・システムに用いる、１つのキャッシュ・メモリとこのキャッシュ・メモリを共有する複数のプロセッサとを含むプロセッサ・モジュールであって、

他のプロセッサ・モジュール内のキャッシュ・メモリにも記憶されている共通データへの各タスクのアクセスを検出する手段と、

アクセスが検出された共通データのアドレスと、この共通データへアクセスしたタスクの識別情報と、同一タスクによる同一データへのアクセス回数とを記憶する記憶手段と

を含むプロセッサ・モジュール。

【請求項１３】　前記アクセスを検出する手段が、

他のプロセッサ・モジュールのキャッシュ・メモリにも記憶されている共通データを書き換えたことによる、前記他のプロセッサ・モジュールのキャッシュ・メモリ内の共通データの無効化の発生を検出する手段と、

前記無効化を発生させたタスクの識別情報と、書き換えられたデータのアドレスと、同一タスクによる同一データでの無効化発生回数とを記憶する手段と

を含む請求項１２のプロセッサ・モジュール。

【請求項１４】　前記アクセスを検出する手段が、

他のプロセッサ・モジュールのキャッシュ・メモリにも記憶されている共通データのうち、前記他のプロセッサ・モジュールの共通データが書き換えられたことによる、キャッシュ・メモリ内の前記共通データの無効化を検出及び記憶する手段と、

前記無効化されたデータへの各タスクのアクセスを検出する手段と

を含む請求項１２のプロセッサ・モジュール。

【請求項１５】　前記共通データの無効化を検出及び記憶する手段が、

データの無効化を検出する手段と、無効化されたデータのアドレスを記憶する手段とを含み、

前記無効化されたデータへの各タスクのアクセスを検出する手段が、

キャッシュ・ミスを検出する手段と、キャッシュ・ミスが発生したデータ・アドレスと前記記憶された無効化されたデータのアドレスとを比較する手段とを含む請求項１４のプロセッサ・モジュール。

【請求項１６】　前記記憶された各アクセス回数又は各無効化発生回数の合計値が所定値を超えると、前記各プロセッサへのタスクの割り当てを、マルチプロセッシングを行っているオペレーティング・システムに要求する手段をさらに含む請求項１２又は請求項１３のプロセッサ・モジュール。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、マルチプロセッサ・システム、マルチプロセッサ・システムに用いるプロセッサ・モジュール及びマルチプロセッシングでのタスクの割り当て方法に関する。

【０００２】

【従来の技術】

ＭＰＵ(Microprocessor Unit)の処理速度を高速化させる方法として、複数のＭＰＵを使用するマルチプロセッサ・システムがある。マルチプロセッサ・システムは、複数のＭＰＵを用いて並列処理を行う。例えば図１０(ａ)に示すように、システム・メモリ２を共有するＭＰＵ１及びＭＰＵ２を用いてマルチプロセッサ・システムを構成する。マルチプロセッシングを行うためには、マルチプロセッサに対応しているＯＳ(Operating System)も必要になる。例えばＯＳに含まれるスケジューラが各ＭＰＵへのタスク又はタスクを細分化したスレッドの割り当てを行う。図１１に示すように、スケジューラが起動すると(Ｓ１００)、スケジューラは例えば優先実行順位に基づいて各タスクを各ＭＰＵに割り当てる(Ｓ１０６)。スケジューラは、タスク間の同期をとる場合等に起動される。例えば所

定時間毎にＯＳがスケジューラを起動する。

【０００３】

図１０(ａ)のＭＰＵ１,ＭＰＵ２は、システム・メモリ２に記憶されるデータの一部が記憶されるキャッシュ・メモリＡ,キャッシュ・メモリＢをそれぞれ備える。以下、キャッシュ・メモリを単にキャッシュとも呼ぶ。マルチプロセッサ・システムでは、システム・メモリ２を共有した状態で並列処理を行うため、各メモリ間のデータの一貫性を保つ必要がある。通常はライト・バック方式のキャッシュ制御を行うので、キャッシュＡ,Ｂ間のデータの一貫性を保つ必要がある。

【０００４】

例えば、ＭＰＵ１及びＭＰＵ２がシステム・メモリ２に記憶されているデータ(DATA_X)を必要としている場合、図１０(ｂ)に示すように、システム・メモリ２に記憶されているDATA_Xをキャッシュ Ａとキャッシュ Ｂにそれぞれ読み出す。ＭＰＵ１とＭＰＵ２が読み出しだけを行っている間は、キャッシュＡ，キャッシュＢの各DATA_Xは同一のデータであり、データの一貫性は保たれている。

【０００５】

しかし、例えばＭＰＵ２がDATA_XをDATA_Xmに更新した場合、キャッシュＢに記憶されているDATA_Xmだけが正しいデータとなり、キャッシュＡに記憶されているDATA_Xは使用してはいけない無効なデータとなる。このときデータの一貫性は保たれていない。通常、図１０(ｃ)に示すように、無効なデータDATA_XをＭＰＵ１が使用しないように、キャッシュＡに記憶されているDATA_Xを無効化(INVALIDATE)する。

【０００６】

DATA_Xが無効化された状態で、ＭＰＵ１がDATA_Xを読み出そうとした場合、キャッシュＡのDATA_Xは無効化されているので、読み出そうとしているDATA_XがキャッシュＡに存在しないと判断される。すなわち、キャッシュ・ミスが発生する。キャッシュＡにDATA_Xが存在しないので、図１０(ｄ)に示すようにキャッシュＢに記憶されているDATA_Xmをキャッシュ Ａに読み出す。この状態では、キャッシュＡとキャッシュＢには、同一のデータ(DATA_Xm)が記憶されているので、デ

ータの一貫性は保たれている。しかし、新たにDATA_Xmが更新されると、更新されなかった方のデータの無効化、さらにはキャッシュ・ミスによるデータの読み出しが同様に起こる。

【０００７】

あるいは、図１０(ｃ)のDATA_Xが無効化された状態で、ＭＰＵ１がDATA_Xを更新しようとした場合、キャッシュＡのDATA_Xは無効化されているので、キャッシュ・ミスとなる。キャッシュＡにDATA_Xmが無いので、図１０(ｄ)に示すようにキャッシュＢに記憶されているDATA_XmをキャッシュＡに読み出し、任意の処理を行った後にDATA_XmをDATA_Xm2に更新する(図示していない)。この状態では、キャッシュＡのDATA_Xm2が正しいデータとなるので、キャッシュＢのDATA_Xmは無効化される(図示していない)。このようなデータの無効化及びキャッシュ・ミスが頻出すると、ＭＰＵの利用率が低下するので、複数のＭＰＵを用いる意味が薄れる。以下、このようなデータの無効化及びキャッシュ・ミスが発生することを、ピンポン現象と呼ぶ。

【０００８】

【発明が解決しようとする課題】

本発明の目的は、ピンポン現象の発生率を低下させることにある。

【０００９】

【課題を解決するための手段】

本発明のマルチプロセッサ・システムは、(ａ)複数のプロセッサ・モジュール内のキャッシュ・メモリに記憶されている共通データへの各タスクのアクセスを検出する手段と、アクセスが検出された共通データのアドレスと、この共通データへアクセスしたタスクの識別情報と、同一タスクによる同一データへのアクセス回数とを記憶する記憶手段とを含むプロセッサ・モジュールと、(ｂ)前記アクセス回数に基づいて、同一の共通データへのアクセス回数が多いタスクを、同一プロセッサ・モジュール内のプロセッサに割り当てる手段とを含む。

【００１０】

本発明のマルチプロセッサ・システムに用いるプロセッサ・モジュールは、他のプロセッサ・モジュール内のキャッシュ・メモリにも記憶されている共通デー

タへの各タスクのアクセスを検出する手段と、アクセスが検出された共通データのアドレスと、この共通データへアクセスしたタスクの識別情報と、同一タスクによる同一データへのアクセス回数とを記憶する記憶手段とを含む。

【００１１】

本発明のマルチプロセッシングでのタスクの割り当て方法は、複数のプロセッサ・モジュール内のキャッシュ・メモリに記憶されている共通データへの各タスクのアクセス状況を監視するステップと、アクセス状況に基づいて、同一の共通データへのアクセス回数が多いタスクを、同一プロセッサ・モジュール内のプロセッサに割り当てるステップとを含む。

【００１２】

【発明の実施の形態】

本発明のマルチプロセッサ・システムは、１つのキャッシュとこのキャッシュを共有する複数のプロセッサとを含むプロセッサ・モジュールを複数用いる。例えば図６に示すように、キャッシュＡとキャッシュＡを共有するＭＰＵ１及びＭＰＵ２とを含むプロセッサ・モジュールＡと、キャッシュＢとキャッシュＢを共有するＭＰＵ３及びＭＰＵ４とを含むプロセッサ・モジュールＢとを用いてマルチプロセサ・システムを構成する。以下、プロセッサ・モジュールを単にモジュールとも呼ぶ。

【００１３】

図６に示すプロセッサ・モジュールを用いた場合、同一モジュール内のプロセッサ間ではキャッシュが共有されるのでピンポン現象は発生しない。例えば図７(ａ),(ｂ),(ｃ)に示すように、キャッシュＡに記憶された同一データ(DATA_X)へのＭＰＵ１とＭＰＵ２のアクセスに関してはピンポン現象は発生しない。しかし、異なるモジュールのプロセッサ間では図１０(ａ)のマルチプロセッサ・システムと同様にピンポン現象が発生する。例えば図８(ａ),(ｂ),(ｃ)に示すように、キャッシュＡとキャッシュＢに記憶された同一データ(DATA_X)へのＭＰＵ１とＭＰＵ３とのアクセスに関しては、図１０(ｂ),(ｃ),(ｄ)と同様にピンポン現象が発生する。ここで、図７(ａ)と図８(ａ)はシステム・メモリ２からのデータ(DATA_X)の読み出し、図７(ｂ)と図８(ｂ)はＭＰＵ１によるデータ更新、図７(ｃ),

図８(ｃ)はそれぞれＭＰＵ２，ＭＰＵ３による更新後のデータ(DATA_Xm)の読み出しを示す図である。

【００１４】

以下、本発明に係るマルチプロセッサ・システム、マルチプロセッサ・システムに用いるプロセッサ・モジュール及びマルチプロセッシングでのタスクの割り当て方法の実施の形態について、図面に基づいて詳しく説明する。図６と同様のＭＰＵ１,ＭＰＵ２,キャッシュＡを含むプロセッサ・モジュールＡと、ＭＰＵ３,ＭＰＵ４,キャッシュＢを含むプロセッサ・モジュールＢとを用いたマルチプロセッサ・システムを例にして説明する。

【００１５】

本発明に係るプロセッサ・モジュールＡは、図１(ａ)に示すように、キャッシュＡ及びプロセッサ・モジュールＢのキャッシュＢに記憶されている同一データのうちの、キャッシュＢ側のデータが書き換えられたことにより無効化されたキャッシュＡ側のデータへの各タスクのアクセスを検出するピンポン検出部１０と、アクセスが検出された無効化されたデータのアドレスと、この無効化されたデータへアクセスしたＭＰＵの識別番号と、同一ＭＰＵによる同一アドレスへのアクセス回数とを含むピンポン情報を生成するピンポン情報生成部２０とを含む。

【００１６】

ピンポン検出部１０は、図１(ｂ)に示すように、キャッシュＡ内のデータの無効化を検出する無効化検出器１２と、無効化されたデータのアドレスを記憶する無効化アドレス記憶部１６と、キャッシュＡ内のキャッシュ・ミスを検出するキャッシュ・ミス検出器１４と、キャッシュ・ミスが検出されたデータのアドレスと無効化されたデータのアドレスとを比較する比較器１８とを含む。無効化の検出は、例えばキャッシュＡのタグ・メモリ内の無効化されたデータ・アドレスを調べる等の方法で行うことができる。

【００１７】

ピンポン情報生成部２０は、図１(ｂ)に示すように、ピンポン情報が記憶されるピンポン情報記憶部２４と、無効化されたデータへアクセスしたＭＰＵとアクセス先のデータ・アドレスをピンポン情報記憶部２４に記憶し、さらに同一ＭＰ

Ｕによる同一アドレスへのアクセス回数をカウントするピンポン情報更新部２２とを含む。モジュールＢもモジュールＡと同様に、ピンポン検出部とピンポン情報生成部とを含む(図示していない)。

【００１８】

スケジューラは、例えばタスク間の同期をとる等の目的でＯＳが起動するが、本発明では図２(ｂ)に示すように、スケジューラが起動すると(Ｓ１００)、ピンポン情報の読み出しを行う(Ｓ１０２)。さらに、ＭＰＵに割り当てられているタスクを識別した後、読み出したピンポン情報に基づいて、ピンポンの発生が最小となるように各タスクを各ＭＰＵに割り当てる(Ｓ１０４,Ｓ１０６)。具体的には、ピンポンの発生状況を解析してグループ分けし(Ｓ１０４)、ピンポンを頻繁に起こしたタスクどうしを同一モジュール内のＭＰＵに割り当てる(Ｓ１０６)。

【００１９】

本発明に係るマルチプロセッシングでは、図２(ａ)に示すように、スケジューラが起動すると(Ｓ１１０)タスクの割り当てを行い(Ｓ１１２)、キャッシュ・ミスが発生すると(Ｓ１２０)ピンポンを検出及び記憶し(Ｓ１２２)、データが無効化されると(Ｓ１３０)無効化されたデータ・アドレスを記憶する(Ｓ１３２)。

【００２０】

次に、このようなプロセッサ・モジュールを備えたマルチプロセッサ・システムでのタスクの割り当て方法について、その作用を説明する。

【００２１】

例えば図８(ａ)と同様に、ＭＰＵ１とＭＰＵ２が必要とするデータがシステム・メモリ２からキャッシュＡに読み出され、ＭＰＵ３とＭＰＵ４が必要とするデータがシステム・メモリ２からキャッシュＢに読み出される。ここで、ＭＰＵ１にはタスク１が割り当てられ、ＭＰＵ２,３,４にはタスク２,３,４がそれぞれ割り当てられていることとする。例えば図８(ｂ)と同様に、キャッシュＡとキャッシュＢの両方に読み出されている同一データのどちらかが更新された場合、更新されなかった側のデータが無効化される。キャッシュ内のデータが無効化されると(Ｓ１３０)、無効化検出器１２がデータの無効化を検出し、無効化されたデータのアドレスを無効化アドレス記憶部１６に記憶する(Ｓ１３２)。

【００２２】

キャッシュ・ミスが発生すると(Ｓ１２０)、キャッシュ・ミスの発生をキャッシュ・ミス検出器１４が検出し、検出されたキャッシュ・ミスのデータ・アドレスと、無効化アドレス記憶部１６に記憶されているアドレスとを比較器１８で比較する。両者が一致すれば、無効化されたデータにアクセスが行われたことが分かる。すなわち、ピンポンが発生したことが分かる。ピンポンの発生が検出されると、ピンポンが検出された無効化されたデータのアドレスと、このアドレスにアクセスしたＭＰＵの識別番号がピンポン情報更新部２２に送られる。

【００２３】

ピンポン情報更新部２２は、無効化されたデータへアクセスしたＭＰＵの識別番号とアクセス先のデータ・アドレスとをピンポン情報記憶部２４に記憶する。ここで、ＭＰＵ１の識別番号は「１」で、ＭＰＵ２,３,４の識別番号はそれぞれ「２」,「３」,「４」とする。ピンポン情報更新部２２は、ＭＰＵ及びアドレスが同一のデータが既に記憶されている場合は、その記憶データのカウント値を１つ増加させる。

【００２４】

ピンポン情報の一例を表１，２に示す。ピンポン情報記憶部２４には、ピンポンを起こしたデータのアドレス(Ping-Pong Address)と、そのアドレスにアクセスしたＭＰＵの識別番号(MPU ID)と、同一ＭＰＵによる同一アドレスへのアクセス回数(Count)とが記憶される。表１はプロセッサ・モジュールＡのピンポン情報であり、表２はプロセッサ・モジュールＢのピンポン情報である。

【表１】

Ping-Pong Address	MPU ID (Task ID)	Count
Address_A	1	100
Address_B	2	8
Address_C	2	60
Address_D	1	3

【表２】

Ping-Pong Address	MPU ID (Task ID)	Count
Address_A	3	80
Address_A	4	20
Address_C	4	50
Address_C	3	10

【００２５】

ＯＳがスケジューラを起動すると(Ｓ１００)、スケジューラはピンポン情報を読み出す(Ｓ１０２)。スケジューラは、ＭＰＵの識別番号(MPU ID)からそのＭＰＵに割り当てられているタスクの識別番号(Task ID)を求め、アクセス回数(Count)に基づいて、各タスクのＭＰＵへの割り当てを行う(Ｓ１０４,Ｓ１０６)。割り当て方法の概要を図３(ａ),(ｂ)に示す。図３(ａ),(ｂ)では、表１及び表２のアドレス(Ping-Pong Address)とタスク(Task ID)との緊密度をアクセス回数(Count)で表したブロック図である。現在のところ、ＭＰＵ１にタスク１が割り当てられ、ＭＰＵ２にタスク２が割り当てられているので、図３(ｂ)に示すようにタスク１とタスク２を同一のグループに分けている。同様に、タスク３とタスク４を同一のグループに分けている。

【００２６】

Address_Aとの緊密度の高さは、タスク１が最も高く、次に高いのがタスク３である。Address_Cとの緊密度の高さは、タスク２が最も高く、次に高いのがタスク４である。スケジューラは、同一アドレスへの緊密度の高いタスクどうしをグループ化する。図３(ａ)に示すように、Address_Aとの緊密度の高いタスク１とタスク３とをグループ化し、Address_Cとの緊密度の高いタスク２とタスク４とをグループ化する(Ｓ１０４)。

【００２７】

スケジューラは、同一グループ内の各タスクを同一モジュール内の各ＭＰＵに割り当てる(Ｓ１０６)。例えば、プロセッサ・モジュールＡのＭＰＵ１にタスク２を、ＭＰＵ２にタスク４を割り当て、プロセッサ・モジュールＢのＭＰＵ３にタスク１を、ＭＰＵ４にタスク３を割り当てる。ＭＰＵ１とＭＰＵ２間及びＭＰ

Ｕ３とＭＰＵ４間ではピンポン現象は起こらない。図３（ａ）のAddress_CへのTask2とTask4のアクセス及びAddress_AへのTask1とTask3のアクセスではピンポンは発生しない。

【００２８】

モジュール間の同一データへのアクセスではピンポン現象が起こる。図３（ｂ）に示すＭＰＵの割り当て変更前のモジュール間のアクセス回数は、

１００＋６０　＝　１６０回

であるが、図３（ａ）に示すＭＰＵの割り当て変更後のモジュール間のアクセス回数は、

２０＋１０　＝　３０回

であり、大幅に減少している。

【００２９】

マルチプロセッシングでは同様の処理を繰り返し行うことが多いので、ピンポンの発生も同様な傾向で繰り返されることが多い。図３（ａ）に示すようにタスクの割り当てを変更することにより、ピンポンの発生は図３（ｂ）の変更前よりも減少する可能性が高い。ピンポン現象の発生率が低下すると、各プロセッサの利用率が向上し、マルチプロセッシングの処理速度も向上する。しかも、同一データへのアクセス回数の多いタスクを同一モジュールのＭＰＵに割り当てることで、モジュール内の各ＭＰＵがアクセスする回数の多いデータがそのモジュール内のキャッシュに記憶されるので、キャッシュのヒット率も向上する。

【００３０】

以上、本発明の一実施例について説明したが、本発明はその他の態様でも実施し得るものである。例えば、マルチプロセッシングに用いるプロセッサ・モジュールは２個に限定はされず、任意数用いることができる。プロセッサ・モジュールに含まれるＭＰＵは２個に限定はされず、任意数用いることができる。

【００３１】

プロセッサへのタスクの割り当てに限定はされず、タスクを細分化したスレッドのプロセッサへの割り当てにも本発明を用いることができる。タスクはスレッドに分けることができるので、１つのタスクから分けられた各スレッドを各モジ

ユールの各プロセッサに割り当てる。

【００３２】

無効化されたデータとキャッシュ・ミスを起こしたデータのアドレスは、そのデータのアドレスを検出及び記憶することもできるが、例えばキャッシュへのデータの読み書きがキャッシュ・ライン毎に行われている場合は、そのライン毎に無効化とキャッシュ・ミスを検出及び記憶することができる。すなわち、同じキャッシュ・ラインに含まれる複数のデータを１つのグループと見なし、このグループ毎に無効化とキャッシュ・ミスを検出及び記憶する。

【００３３】

図４に示すように、ピンポン情報更新部３２で各アクセス回数(Count)の総合計を求め、ピンポン情報内のアクセス回数の合計値が所定値を超えると、割り込みにより、タスクの割り当てをマルチプロセッシングを行っているオペレーティング・システムに要求することもできる。アクセス回数の合計値と比較する前記所定値を変更することにより、タスクの割り当てを最適化する頻度を調整することができる。このような割り込みによるタスクの割り当ての実行は、上述したスケジューラの起動の代わりに用いることもできるし、スケジューラの起動とこの割り込みとを併用することもできる。割り込みの発生は、例えば単位時間あたりのピンポン発生回数が所定値を超えた場合に発生させることもできる。

【００３４】

無効化されたデータへのアクセスは、書き込みと読み出しの両方を検出する以外に、書き込みアクセスのみを検出してもよいし、読み出しアクセスのみを検出してもよい。ピンポンの発生回数に基づいてタスクの割り当てを行う以外に、ピンポンが発生する前段階のデータの無効化に基づいてタスクの割り当てを行ってもよい。例えば図５に示すように、プロセッサ・モジュールＢのキャッシュ・メモリＢにも記憶されているキャッシュ・メモリＡ内の共通データを書き換えたことによる、キャッシュ・メモリＢ内の共通データの無効化の発生を検出する無効化検出部４０、無効化を発生させたタスクの識別情報と、書き換えられたデータのアドレスと、同一タスクによる同一データでの無効化発生回数とを記憶する無効化情報記憶部４２とを含ませる。無効化情報の一例を表３に示す。

【表３】

Invalidate Address	MPU ID (Task ID)	Count
Address_A	1	100
Address_B	2	8
Address_C	2	60
Address_D	1	3

【００３５】

無効化情報記憶部４２に記憶されているアドレス(Invalidate Address)とＭＰＵの識別番号(MPU ID)と無効化発生回数(Count)に基づいて、上述の実施形態と同様にタスクをＭＰＵに割り当てることができる。ピンポンと無効化の両方を検出してもよい。さらに、データの無効化の代わりに、複数のキャッシュに記憶された同一データへの読み出しアクセスに基づいてタスクの割り当てを行うこともできる。複数のキャッシュに記憶された共通データへの読み出しアクセスの検出とデータの無効化の検出と無効化されたデータへのアクセスの検出のいずれか１つ又は複数を組み合わせることができる。

【００３６】

複数のキャッシュに記憶されている共通データのいずれかが更新されると、更新されたデータ以外のデータは無効化され、更新されたデータは排他状態から共有状態に遷移する。この排他状態から共有状態への遷移を、無効化の代わりに検出することもできる。共有状態のデータへの各タスクのアクセス回数に基づいてタスクの割り当てを行うことができる。

【００３７】

ライト・バック方式のキャッシュ制御を例にして説明したが、ライト・スルー方式のキャッシュ制御でも本発明を用いることができる。さらにＤＭＡ(Direct Memory Access)等のＭＰＵを介さず直接システム・メモリ２にアクセス可能な場合は、キャッシュ及びシステム・メモリ２間でデータの一貫性を保つ必要がある。例えば図９(ａ)に示すようにキャッシュＡ,Ｂにシステム・メモリ２のDATA_Xが読み出された状態でシステム・メモリ２のDATA_XがDATA_Xmに更新されると、図９(ｂ)に示すようにキャッシュＡ,ＢのDATA_Xを無効化する。キャッシュＢからD

ATA_Xを読み出そうとするとキャッシュ・ミスとなり、図９（ｃ）に示すようにシステム・メモリ２からキャッシュＢへDATA_Xmを読み出す。

【００３８】

以上、本発明は特定の実施例について説明されたが、本発明はこれらに限定されるものではない。その他、本発明はその趣旨を逸脱しない範囲で当業者の知識に基づき種々なる改良，修正，変形を加えた態様で実施できるものである。

【００３９】

【発明の効果】

本発明のプロセッサ・モジュールは、ピンポンを起こしたアドレスと、そのアドレスにアクセスしたタスクと、同一タスクによる同一アドレスへのアクセス回数とを監視することができる。このようなプロセッサ・モジュールを用いたマルチプロセッサ・システムは、同一データへのアクセス回数の多いタスクを同一モジュール内のプロセッサに割り当てることができる。

【００４０】

本発明のマルチプロセッシングでのタスクの割り当て方法は、ピンポンを起こしたアドレスと、そのアドレスにアクセスしたタスクと、同一タスクによる同一アドレスへのアクセス回数とを監視し、同一データへのアクセス回数の多いタスクを同一モジュール内のプロセッサに割り当てる。ピンポンによるモジュール間のアクセス数が最小となるように各プロセッサへタスクを割り当てることができる。

【図面の簡単な説明】

【図１】

同図（ａ）は本発明に係るプロセッサ・モジュールの一構成例を示すブロック図であり、同図（ｂ）は同図（ａ）に示すピンポン検出部とピンポン情報生成部の一構成例を示すブロック図である。

【図２】

同図（ａ）は本発明に係るマルチプロセッシングのタスクの割り当て手順の概略例を示す図であり、同図（ｂ）は同図（ａ）に示すタスクの割り当て（Ｓ１１２）手順の詳細例を示す図である。

【図３】

同図(ａ)はタスクのグループ化を最適化した例を示すブロック図であり、同図(ｂ)はタスクのグループ化を最適化する前の例を示すブロック図である。

【図４】

本発明に係るプロセッサ・モジュールのピンポン情報生成部の他の構成例を示すブロック図である。

【図５】

本発明に係るプロセッサ・モジュールの他の構成例を示すブロック図である。

【図６】

本発明に係るマルチプロセッサ・システムの基本構成を示すブロック図である。

【図７】

同図(ａ),(ｂ),(ｃ)は図６に示すキャッシュ内のデータ更新の一例を示すブロック図である。

【図８】

同図(ａ),(ｂ),(ｃ)は図６に示すキャッシュ内のデータの更新及び無効化の一例を示すブロック図である。

【図９】

同図(ａ),(ｂ),(ｃ)は図６に示すキャッシュ内のデータの更新及び無効化の他の例を示すブロック図である。

【図１０】

同図(ａ)は従来のマルチプロセッサの一構成例を示すブロック図であり、同図(ｂ),(ｃ),(ｄ)はキャッシュ内のデータの更新及び無効化の一例を示すブロック図である。

【図１１】

従来のマルチプロセッシングのタスクの割り当て手順の一例を示す図である。

【符号の説明】

２：システム・メモリ

１０：ピンポン検出部

１２：無効化検出器

１４：キャッシュ・ミス検出器

１６：無効化アドレス記憶部

１８：比較器

２０，３０：ピンポン情報生成部

２２，３２：ピンポン情報更新部

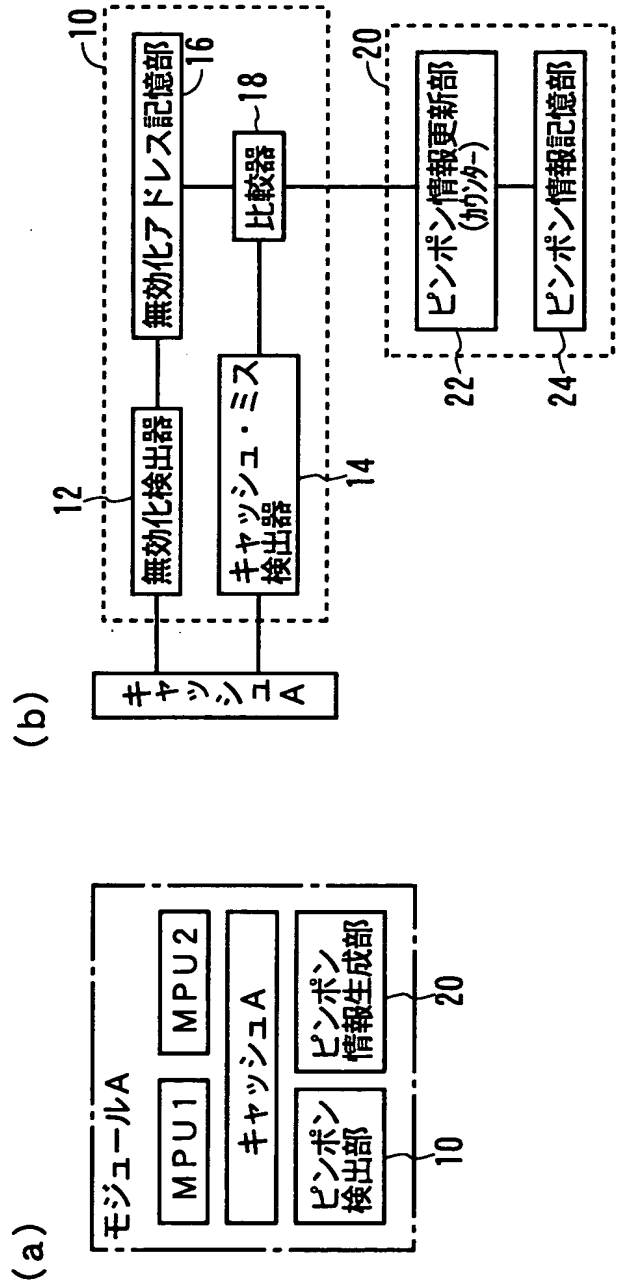
２４：ピンポン情報記憶部

４０：無効化検出部

４２：無効化情報記憶部

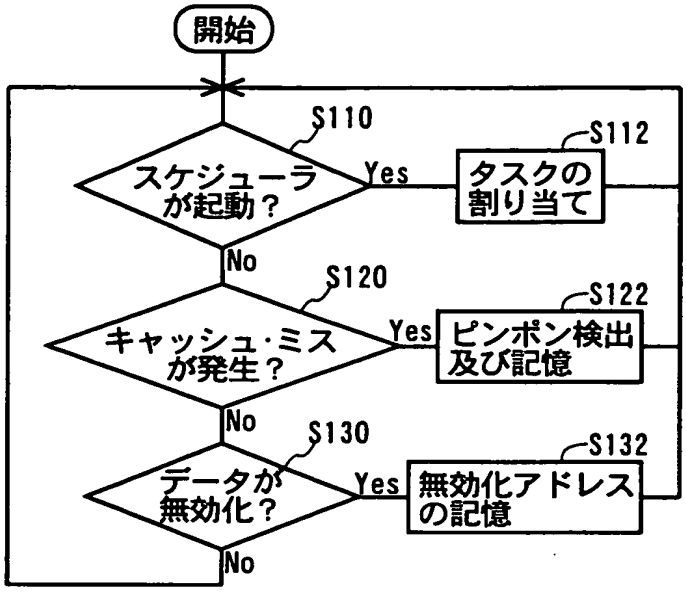
【書類名】図面

【図１】

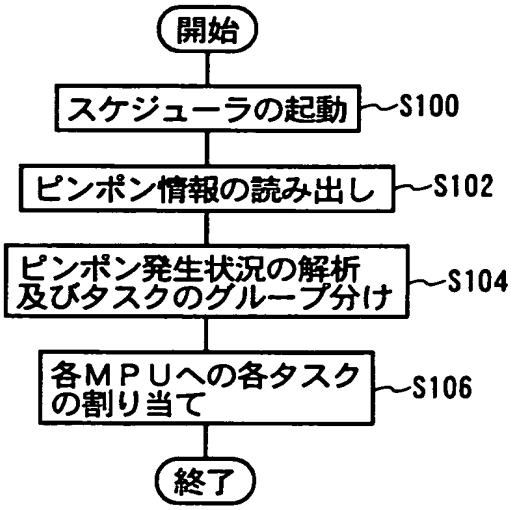

(b)
10
無効化アドレス記憶部
16
比較器
18
20
ピンポン情報更新部
(カウンター)
ピンポン情報記憶部
12
無効化検出器
キャッシュ・ミス
検出器
14
22
24
キャッシュA
(a)
モジュールA
MPU1
MPU2
キャッシュA
ピンポン
検出部
ピンポン
情報生成部
10
20

【図２】

（ａ）

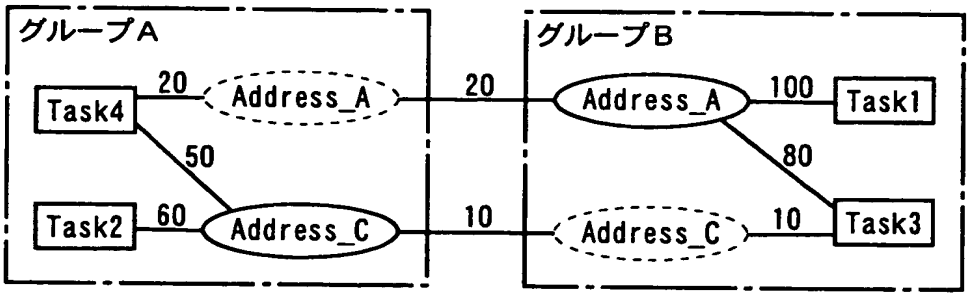

開始
S110
スケジューラが起動？
Yes
S112
タスクの割り当て
No
S120
キャッシュ・ミスが発生？
Yes
S122
ピンポン検出及び記憶
No
S130
データが無効化？
Yes
S132
無効化アドレスの記憶
No

（ｂ）

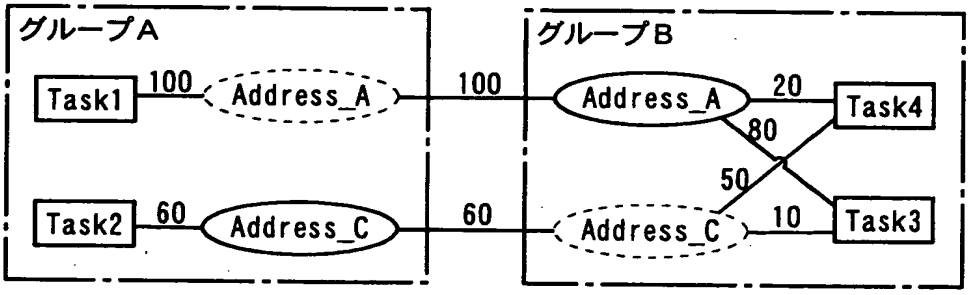

開始
スケジューラの起動
S100
ピンポン情報の読み出し
S102
ピンポン発生状況の解析及びタスクのグループ分け
S104
各ＭＰＵへの各タスクの割り当て
S106
終了

【図３】

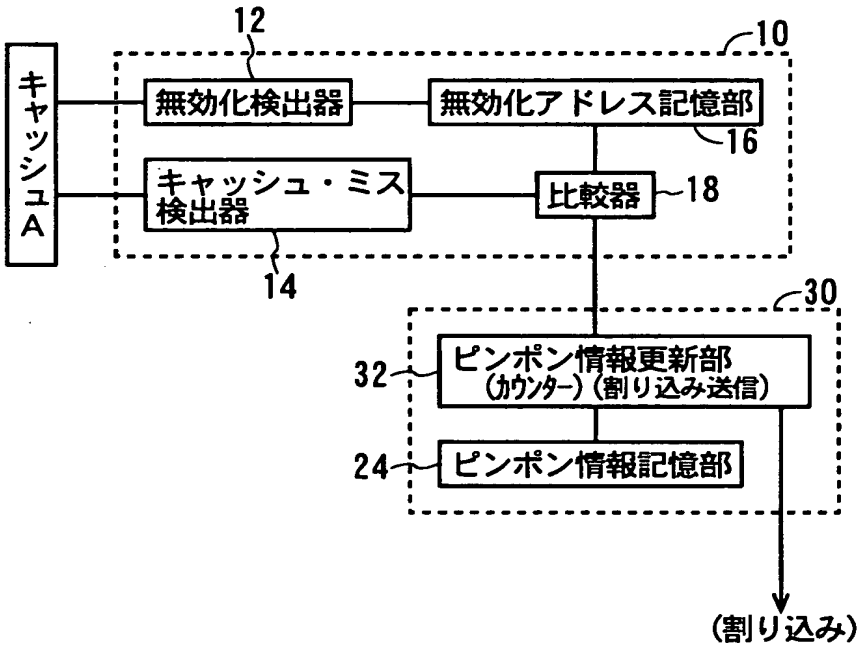
(a)


グループA
グループB
Task4
20
Address_A
20
Address_A
100
Task1
50
80
Task2
60
Address_C
10
Address_C
10
Task3

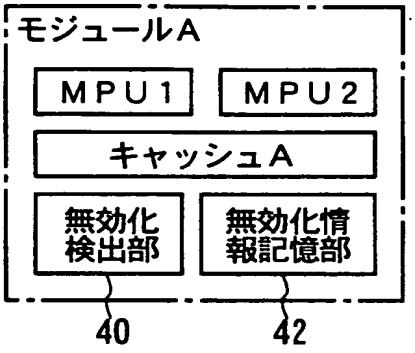
(b)


グループA
グループB
Task1
100
Address_A
100
Address_A
20
Task4
80
50
Task2
60
Address_C
60
Address_C
10
Task3

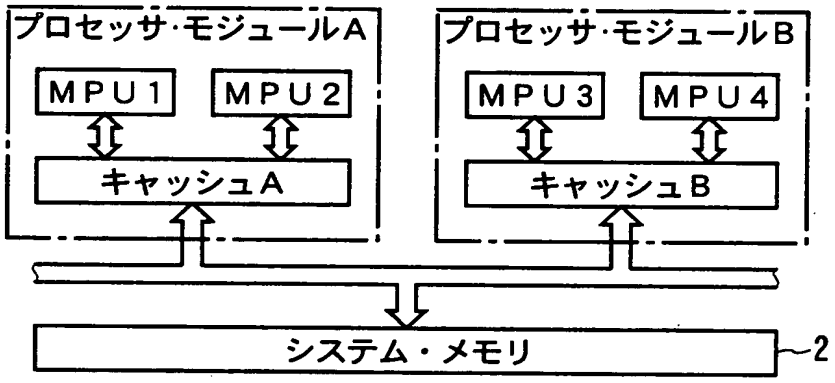
【図４】


12
10
キャッシュA
無効化検出器
無効化アドレス記憶部
16
キャッシュ・ミス検出器
比較器
18
14
30
32
ピンポン情報更新部
(カウンター)(割り込み送信)
24
ピンポン情報記憶部
(割り込み)

【図５】

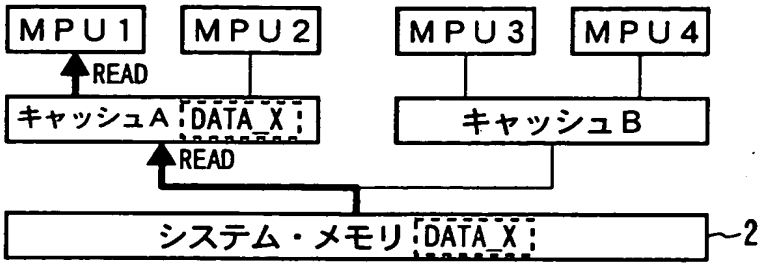

モジュールA
MPU1
MPU2
キャッシュA
無効化
検出部
無効化情
報記憶部
40
42

【図６】

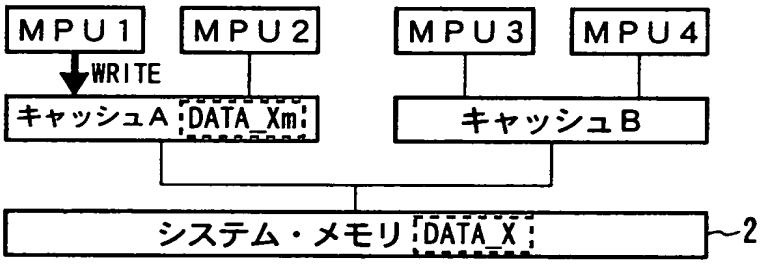

プロセッサ･モジュールA
MPU1
MPU2
キャッシュA
プロセッサ･モジュールB
MPU3
MPU4
キャッシュB
システム・メモリ
2

【図７】

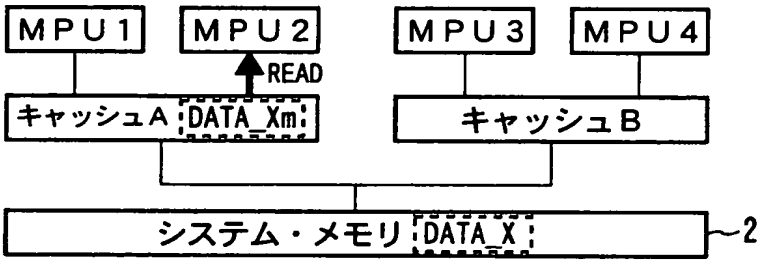
(a)


MPU1
MPU2
MPU3
MPU4
READ
キャッシュA
DATA_X
キャッシュB
READ
システム・メモリ
DATA_X
2

(b)

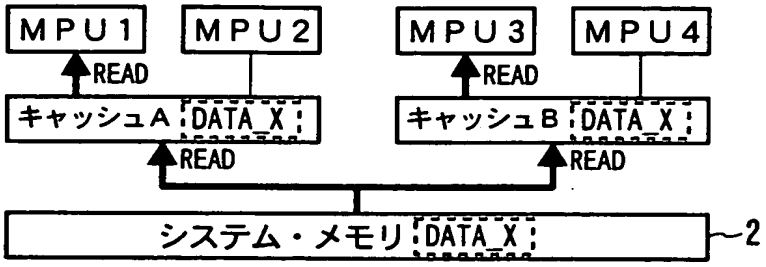

MPU1
MPU2
MPU3
MPU4
WRITE
キャッシュA
DATA_Xm
キャッシュB
システム・メモリ
DATA_X
2

(c)

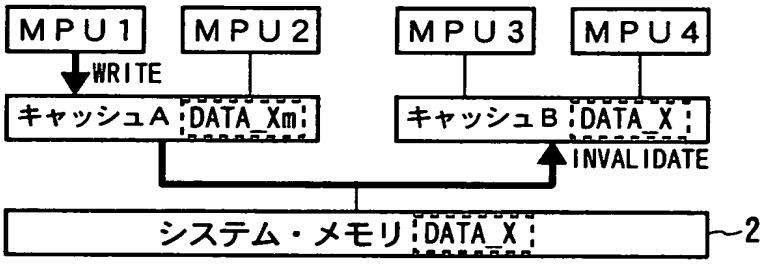

MPU1
MPU2
MPU3
MPU4
READ
キャッシュA
DATA_Xm
キャッシュB
システム・メモリ
DATA_X
2

【図８】

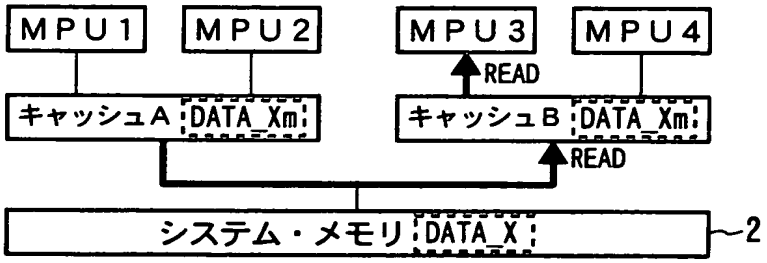
(a)


MPU1
MPU2
MPU3
MPU4
READ
READ
キャッシュA DATA_X
キャッシュB DATA_X
READ
READ
システム・メモリ DATA_X
2

(b)

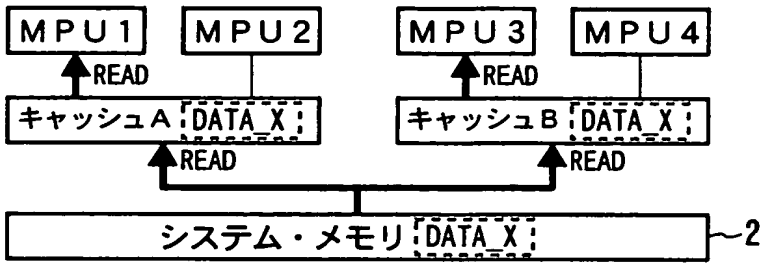

MPU1
MPU2
MPU3
MPU4
WRITE
キャッシュA DATA_Xm
キャッシュB DATA_X
INVALIDATE
システム・メモリ DATA_X
2

(c)

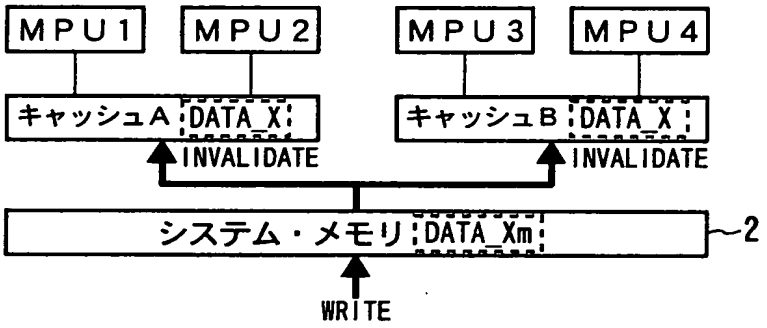

MPU1
MPU2
MPU3
MPU4
READ
キャッシュA DATA_Xm
キャッシュB DATA_Xm
READ
システム・メモリ DATA_X
2

【図９】

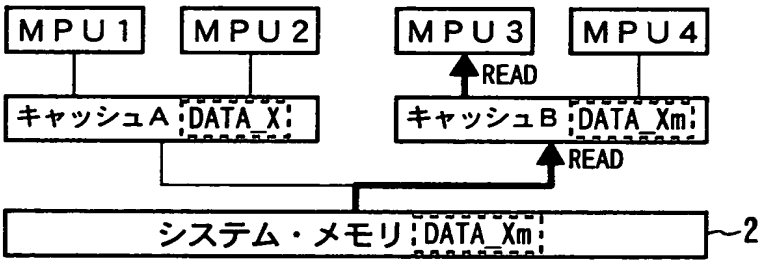
(a)


MPU1
MPU2
MPU3
MPU4
READ
READ
キャッシュA DATA_X
キャッシュB DATA_X
READ
READ
システム・メモリ DATA_X
2

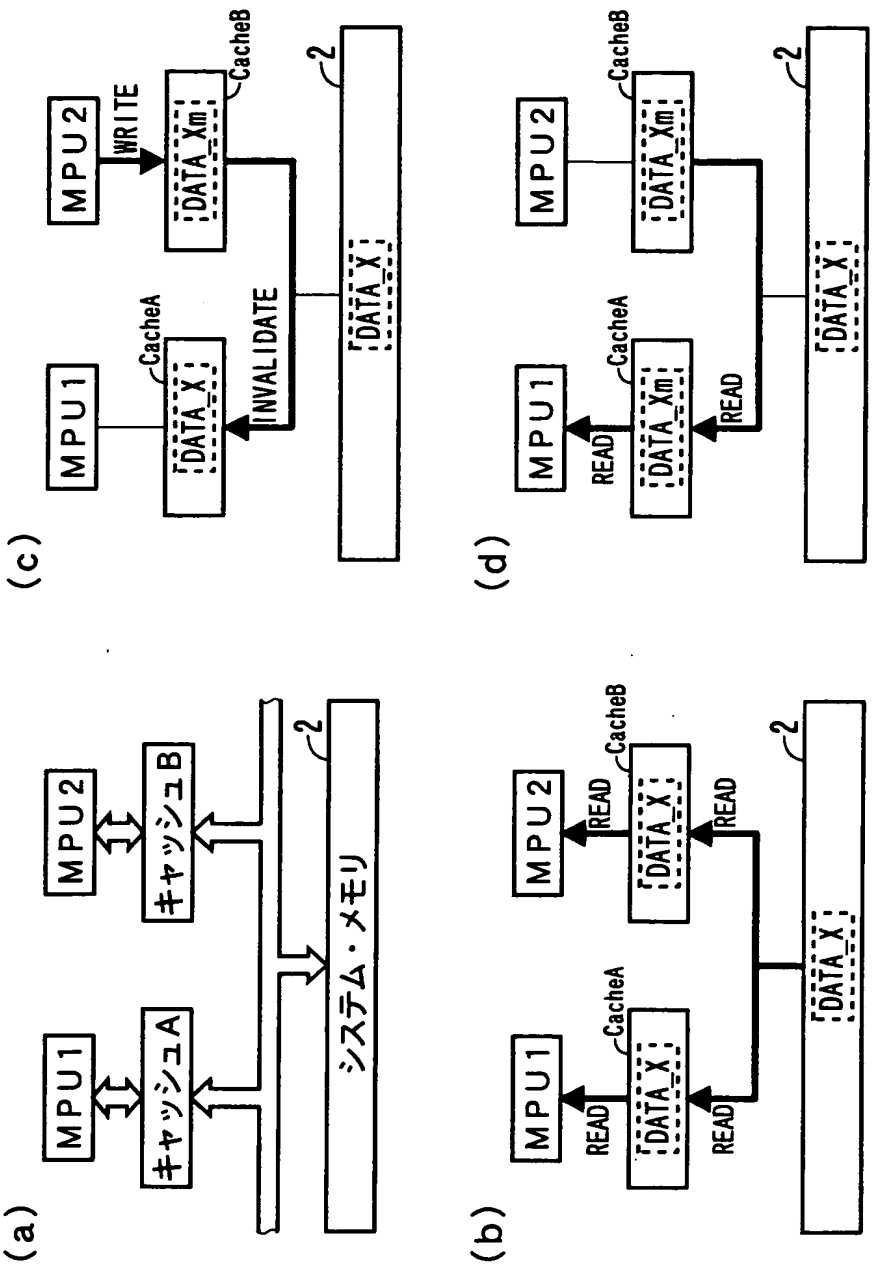
(b)


MPU1
MPU2
MPU3
MPU4
キャッシュA DATA_X
キャッシュB DATA_X
INVALIDATE
INVALIDATE
システム・メモリ DATA_Xm
2
WRITE

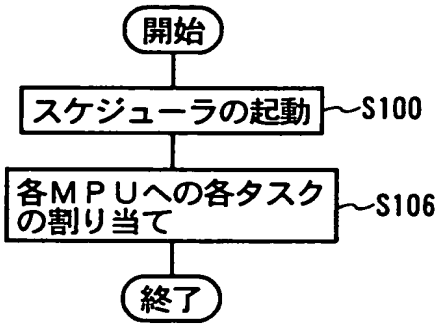
(c)


MPU1
MPU2
MPU3
MPU4
READ
キャッシュA DATA_X
キャッシュB DATA_Xm
READ
システム・メモリ DATA_Xm
2

【図１０】


(a)
MPU1
キャッシュA
MPU2
キャッシュB
システム・メモリ
2
(b)
MPU1
READ
CacheA
DATA_X
READ
MPU2
READ
CacheB
DATA_X
READ
DATA_X
2
(c)
MPU1
CacheA
DATA_X
INVALIDATE
MPU2
WRITE
CacheB
DATA_Xm
DATA_X
2
(d)
MPU1
READ
CacheA
DATA_Xm
READ
MPU2
CacheB
DATA_Xm
DATA_X
2

【図１１】



【書類名】　　要約書

【要約】

【課題】　ピンポン現象の発生率を低下させる。

【解決手段】　他のプロセッサ・モジュール内のキャッシュ・メモリにも記憶されている共通データへの各タスクのアクセスを検出する手段１０と、アクセスが検出された共通データのアドレスと、この共通データへアクセスしたタスクの識別情報と、同一タスクによる同一データへのアクセス回数とを記憶する記憶手段２０とを含むプロセッサ・モジュールを複数用い、記憶手段２０に記憶されたアドレスとタスクの識別情報とアクセス回数とに基づいて、同一アドレスへのアクセス回数の多いタスクを同一プロセッサ・モジュール内のプロセッサに割り当てる。

【選択図】　　図１（ａ）

認定・付加情報

特許出願の番号	特願２０００－２３６７４０
受付番号	５００００９９４３９４
書類名	特許願
担当官	塩崎　博子　　１６０６
作成日	平成１２年　９月２１日

＜認定情報・付加情報＞

【提出日】　平成12年 8月 4日

【特許出願人】

【識別番号】　390009531

【住所又は居所】　アメリカ合衆国１０５０４、ニューヨーク州　アーモンク　（番地なし）

【氏名又は名称】　インターナショナル・ビジネス・マシーンズ・コーポレーション

【代理人】

【識別番号】　100086243

【住所又は居所】　神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　大和事業所内

【氏名又は名称】　坂口　博

【復代理人】　申請人

【識別番号】　100094248

【住所又は居所】　滋賀県大津市粟津町４番７号　近江鉄道ビル５Ｆ　楠本特許事務所

【氏名又は名称】　楠本　高義

【選任した代理人】

【識別番号】　100091568

【住所又は居所】　神奈川県大和市下鶴間１６２３番地１４　日本アイ・ビー・エム株式会社　大和事業所内

【氏名又は名称】　市位　嘉宏

次頁無

出　願　人　履　歴　情　報

識別番号　　〔３９０００９５３１〕

１．変更年月日　　２０００年　５月１６日
［変更理由］　名称変更
住　所　　アメリカ合衆国１０５０４、ニューヨーク州　アーモンク　（番地なし）
氏　名　　インターナショナル・ビジネス・マシーンズ・コーポレーション